Subject: Re: problem with a processor (software-wise)
Posted by miree on Wed, 13 Aug 2014 16:10:33 GMT
View Forum Message <> Reply to Message

Hi Tayfun,

I never was looking in detail to the pile-up correcting readout of the MUSIC. First of all: Only one of the two MUSICS was read out with the pile-up correction module. The other one was read out in the conventional way. The pile-up correction readout is done with a SIS3302 digitizer from Struck with a special firmware from KVI Groningen. This digitizer has 8 channels for incoming signals.
The unpacker delivers for each of the channels an amplitude (having indices from 0 to 7), and time information (having indices ranging from 8 to 15, where (index-8) will tell you the module channel). Or, in other words: you get 16 output channels (or indices), where the first half contains amplitude information, the second half contains time information.
This module is multi-hit capable, that means that you can have several hits in each channel. You need to select somehow the "good" hit if there are is more than one hit.

One thing that you could do in addition: check the correlation of the SIS-MUSIC amplitudes with the amplitudes from the classical music readout of the other MUSIC. I'have seen an experiment, where there was no correlation at all between the SIS and classical readout.

Regarding the error message in the implementation of your processor. I've written the code as I think it should work in a verbose manner, to point out what is causing the problem in your implementation:

```
// check first if both arrays have the same number of entries
if (input_array_size(value1) == input_array_size(value2))
{
  for (int i = 0; i < input_array_size(value1); ++i)
  {
    // At this point, your code has a problem, because you use "index" instead of "i".
    // The second parameter in the functions input_array_index/input_array_value
    // must not exceed the size of the array (i.e. the number of entries of the array)
    // Otherwise you'll get an assertion failure.
    int    index_of_ith_entry_in_value1 = input_array_index(value1, i);
    double value_of_ith_entry_in_value1 = input_array_value(value1, i);
    int    index_of_ith_entry_in_value2 = input_array_index(value2, i);
    double value_of_ith_entry_in_value2 = input_array_value(value2, i);

    // eventually, you want to check if the indices are the same
    if (index_of_ith_entry_in_value1 == index_of_ith_entry_in_value2)
    {
      if (condition_valid(value_of_ith_entry_in_value2, gate, index_of_ith_entry_in_value1))
      {
        fill_output_array(GatedValue, index_of_ith_entry_in_value2,
value_of_ith_entry_in_value1)
      }
    }
  }
}
```

Just keep in mind the following: Index of an array entry index_of_ith_entry_in_value1 is something different than the entry number i. Mixing up these two things causes your code to stop with an assertion failure.
